# Hands Free Video Control Rehabilitation

**A PROJECT REPORT**

***Submitted by***

## Rajesh kanna T (REG.No:921722108307)

**Mothan Raj M (REG.No:** **921722108068)**

**Munish Karthikeyan B (REG.No:** **921722108070)**

**Vignesh A (REG.No:** **921722108309)**

***for the Course***

## 21UIT607 – PRODUCT DEVELOPMENT PROJECT

**IN**

**B.Tech. INFORMATION TECHNOLOGY**

# SETHU INSTITUTE OF TECHNOLOGY

An Autonomous Institution | Accredited with ‘A++’ Grade by NAAC

**PULLOOR, KARIAPATTI-626 115.**

****

## ANNA UNIVERSITY: CHENNAI 600 025

**APRIL 2025**

# SETHU INSTITUTE OF TECHNOLOGY

An Autonomous Institution | Accredited with ‘A’ Grade by NAAC

**PULLOOR, KARIAPATTI-626 115**

# ANNA UNIVERSITY: CHENNAI 600 025

## BONAFIDE CERTIFICATE

Certified that this project report **“Hands Free Video Control Rehabilitation”** is the bonafide work of **“Rajesh Kanna T (921722108307), Vignesh A (921722108309), Munish Karthikeyan B (921722108070), Mothan Raj (921722108068),”** who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**Dr.M.Poomani @ Punitha B.Tech.,M.E.,Ph.D., [Guide Name]**

**Head of the Department [Supervisor / Designation]** Department of Information Technology Department of Information Technology Sethu Institute of technology Sethu Institute of technology

Pullor, Kariapatti-626 115. Pullor, Kariapatti-626 115.

Submitted for the project viva-vice conducted on\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ABSTRACT**

Hands-Free Video Control for Rehabilitation is a smart video player that uses AI technology to make video watching easier and more accessible. It works by detecting eye movements to automatically play or pause the video when the user is looking at or away from the screen. It also supports hand gestures, allowing users to adjust volume, skip, or rewind without touching any buttons.This system is especially helpful for people with disabilities, patients in rehabilitation, or those in hands-busy environments like hospitals, kitchens, or workshops. By using computer vision and AI, the video player provides a smooth, touch-free experience. It improves ease of use, hygiene, and accessibility, making it useful for medical care, education, and assistive technology.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Title** | **Page No.** |
|  | Abstract | 1 |
| 1 | Problem Statement | 3 |
| 2 | Existing Solution | 4 |
| 3 | Proposed Solution | 5 |
| 4 | Need for the Product | 5 |
| 5 | Market Analysis | 6 |
| 6 | Marketing Channels | 8 |
| 7 | Financial Estimation | 8 |
| 8 | Product Implementation | 9 |
| 9 | Conclusion | 9 |
| 10 | References | 10 |
|  | COs & POs Mapping | 11 |

**CHAPTER 1**

**PROBLEM STATEMENT**

Current video players depend on traditional input methods like mouse clicks, keyboard shortcuts, or touchscreens, which can be difficult for people with physical disabilities or those needing hands-free control. Voice commands can be imprecise and unreliable, and in many situations, users need a more efficient way to interact with video content while multitasking or in environments where physical interaction is not feasible.

**CHAPTER 2**

**EXISTING SOLUTION**

Several hands-free video control technologies exist, but each has some problems. **Voice-controlled systems** like Alexa, Google Assistant, and Siri let users control videos by speaking, but they do not work well in noisy places and lack precise control over video playback.

**Gesture-based systems**, such as Microsoft Kinect, Google Soli, and Samsung Smart TVs, use sensors to detect hand movements for controlling playback and volume. However, they need special devices, work best in good lighting, and are not commonly used for video playback.

**Eye-tracking technology**, like Tobii Eye Tracker and Windows Eye Control, allows users to control videos by looking at the screen. While it helps people with disabilities, it is expensive, needs setup for each user, and may not work well in low light or for people wearing glasses.

**Traditional video players**, such as VLC and Windows Media Player, require a keyboard or mouse, which can be hard to use for people with limited mobility. Similarly, mobile touch controls, like in YouTube and Netflix, need physical interaction, making them difficult to use when hands are busy.

Even though these options exist, none combine **eye-tracking and hand gestures** to give a fullyhands-free experience. The proposed system solves this problem using AI-powered computer vision, making video control easier, more accessible, and useful for rehabilitation.

**CHAPTER 3**

**PROPOSED SOLUTION**

The **Hands-Free Video Control for Rehabilitation** system is a smart video player that allows users to control videos without touching the screen. It uses AI technology to recognize eye movements and hand gestures, making it easier for people with disabilities or those who have their hands busy to interact with videos. The system works with a regular webcam or built-in camera, so no extra devices are needed.

The **eye-tracking feature** helps control video playback. If the user looks away, the video pauses automatically. When they look back at the screen, the video starts playing again. This ensures that users do not miss any part of the video. Along with eye-tracking, the system also has hand gesture controls to adjust volume, skip forward or backward, and pause or play the video. Instead of using a remote or touchscreen, users can simply move their hands to control playback.

The system uses **computer vision and AI models** to detect faces and hand movements accurately. It is designed to work on different devices, including computers, tablets, and smart TVs. Since it does not rely on voice commands, it works well in noisy environments, where voice control might not be effective.

This solution is especially helpful for people with mobility challenges and those in rehabilitation centers, as it allows them to watch videos without needing help. It is also useful for people in hands-busy situations like cooking, exercising, or working in a medical setting.

By combining **eye-tracking and hand gesture control**, this system offers a simple, smart, and easy way to interact with videos. It makes video watching more accessible, convenient, and touch-free for everyone.

**Sustainable Development Goals (SDGs)**

1. **SDG 3 – Good Health and Well-being**
   * Helps people in **rehabilitation** by allowing them to control videos **without help.**
   * Reduces stress for those with **mobility challenges** by making video interaction **easy.**
2. **SDG 4 – Quality Education**
   * Supports **inclusive learning** for students with **disabilities**.
   * Helps people in **rehabilitation centers** access educational videos **easily**.
3. **SDG 9 – Industry, Innovation, and Infrastructure**
   * Uses **AI and smart technology** to improve accessibility.
   * Encourages **new assistive technologies** for people with special needs.
4. **SDG 10 – Reduced Inequalities**
   * Gives **equal access** to video control for people with disabilities.
   * Makes technology more **inclusive for all users.**
5. **SDG 11 – Sustainable Cities and Communities**
   * Supports **smart and accessible technology** for everyone.
   * Promotes **universal design**, making video control **easy for all.**

**CHAPTER 4**

**NEED FOR THE PRODUCT**

1. **Helps People with Disabilities** – Individuals with mobility challenges can control videos using **eye movements and hand gestures** instead of buttons or touchscreens.
2. **Useful in Rehabilitation** – Patients recovering from injuries or surgeries can watch videos **without needing help** from others.
3. **Convenient for Hands-Busy Situations** – Useful for **cooks, gym users, and medical professionals** who need to watch videos while their hands are occupied.
4. **No Need for Touch or Voice Commands** – Works well in **noisy environments** where voice control may not be effective and avoids the need for physical contact.
5. **Enhances Accessibility** – Makes video watching **easier and more inclusive** for people who find traditional controls difficult to use.
6. **Works with Regular Cameras** – No need for special hardware, as the system works with **built-in webcams or device cameras.**
7. **Smarter and More Advanced** – **Uses AI, eye-tracking, and hand gesture recognition** to provide a modern, easy, and touch-free way to control videos.
8. **Reduces Dependency on Others – People with mobility issues can independently control videos without needing assistance.**

**CHAPTER 5**

**MARKET ANANLYSIS**

### ****1. Growing Demand for Accessibility Solutions****

* The number of people with **disabilities and mobility impairments** is increasing, creating a need for **hands-free technology.**
* Rehabilitation centers, hospitals, and elderly care facilities require **assistive technologies** for patient support.

### ****2. Expanding AI and Gesture-Control Market****

* The **AI-powered video control** market is growing, with advancements in **computer vision and gesture recognition.**
* Major tech companies are investing in **touch-free interaction technologies,** showing a rising demand for **smart accessibility solutions.**

### ****3. Existing Solutions Have Limitations****

* **Voice-controlled systems** (like Alexa, Google Assistant) struggle in **noisy environments**.
* **Touchscreens and remotes** are **inconvenient for people with disabilities or hands-busy users.**
* **Gesture-based controls** exist in gaming but are **not widely used for video playback.**

### ****4. Potential Users and Industries****

* **People with disabilities** who need easy access to media.
* **Rehabilitation centers and hospitals** for patient engagement.
* **Fitness, cooking, and medical professionals** who need hands-free video control.
* **Smart home users** looking for innovative, touch-free media solutions.

### ****5. Competitive Advantage****

* Combines **eye-tracking and hand gestures** for a **seamless, hands-free experience.**
* Works with **regular webcams and built-in cameras,** eliminating the need for extra hardware.
* More **accurate and accessible** compared to **voice and traditional gesture-based controls.**

**CHAPTER 6 MARKETING CHANNELS**

1. **Social Media Marketing** – Promote the product through **Facebook, Instagram, LinkedIn, and Twitter** using **videos, ads, and influencer collaborations** to reach tech enthusiasts, healthcare professionals, and accessibility advocates.
2. **Digital Advertising** – Use **Google Ads, YouTube Ads, and social media ads** to target users searching for **gesture-based technology, accessibility tools, and AI-powered solutions.**
3. **Healthcare & Rehabilitation Centers** – Partner with **hospitals, therapy centers, and disability support organizations** to introduce the system to people who need hands-free solutions.
4. **Tech and Accessibility Forums** – Engage with **Reddit, Quora, and accessibility blogs** to share the benefits of the system and build trust among potential users.
5. **Influencer & Expert Endorsements** – Collaborate with **tech bloggers, YouTubers, and accessibility advocates** to showcase real-world applications and attract users.
6. **Partnerships with Smart Device Companies** – Work with **hardware manufacturers** to integrate the system into **smart TVs, computers, and AR/VR devices.**
7. **SEO & Content Marketing** – Publish blogs, tutorials, and case studies on **Google-friendly topics** related to **AI video control, accessibility tech, and hands-free computing** to attract organic traffic.
8. **B2B Outreach & Enterprise Sales** – Offer the product to **corporate offices, educational institutions, and industries** where hands-free control improves productivity and accessibility.

**CHAPTER 7**

**FINANCIAL ESTIMATION**

### The estimated budget includes development, hardware, and marketing costs, ensuring a cost-effective implementation using open-source AI, affordable hardware, and free promotional strategies like social media and word-of-mouth.

### ****Development Costs** – AI & software development: ₹3500- ₹5000**

### ****Hardware Costs** – Web Camera & testing: ₹5,000 - ₹10,000**

### ****Marketing & Promotion** – Minimal ads & promotions: ₹30,000 - ₹50,000**

### ****Colaboration Costs** – VLC,MX Media Players ₹15,000 - ₹30,000 per year (Future Implementation)**

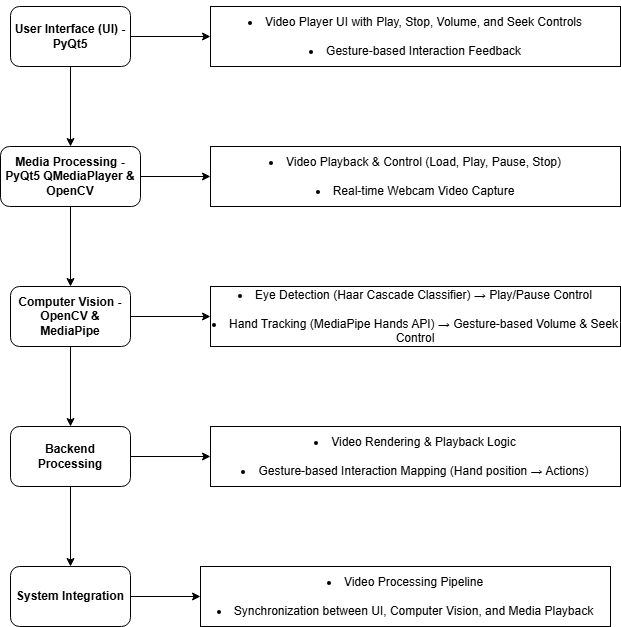
### ****Total Estimated Budget** – ₹50,000 - ₹60,000 (initial).**

### **Costs can be minimized by using open-source AI models, low-cost hardware, and free marketing methods like social media and word-of-mouth.**

**CHAPTER 7**

**PRODUCT IMPLEMENTATION**

**SYSTEM ARCHITECTURE**

****

#### MODULES

#### ****1. User Interface (UI) - PyQt5****

* **Video Display:** Uses QVideoWidget for rendering.
* **Controls:** Play, Stop, Open File, Volume (QSlider), and Seek (QSlider).
* **Event Handling:** UI elements interact with QMediaPlayer.

#### ****2. Media Processing - PyQt5 QMediaPlayer & OpenCV****

* **QMediaPlayer:** Loads, plays, pauses, stops video.
* **QVideoWidget:** Displays video content.
* **Webcam Feed:** Captures frames for gesture recognition using OpenCV.

#### ****3. Computer Vision - OpenCV & MediaPipe****

* **Eye Detection (Haar Cascade Classifier)**
  + **Eyes detected:** Play video.
  + **Eyes closed:** Pause video.
* **Hand Tracking (MediaPipe Hands API)**
  + **Volume Control:** Hand movement in the left region.
  + **Seek Control:** Hand movement in the right region.

#### ****4. Gesture-Based Control Mapping****

* **Volume Control (Left Box) →** Move hand up/down to increase/decrease volume.
* **Seek Control (Right Box) →** Move hand left/right to rewind/forward video.

#### ****5. Backend Processing****

* **Synchronizes UI, Media Playback, and Computer Vision.**
* **Handles gesture-to-action mapping and updates UI accordingly.**
* **Manages resources (OpenCV & PyQt5) for smooth execution.**

#### ****6. System Integration****

* QTimer**executes**process\_frame()**every 100ms** for real-time gesture recognition.
* **Signal-Slot mechanism ensures UI updates in sync with gesture inputs**.

### ****Hardware Components:****

1. **Web Camera** – Captures eye and hand movements.
2. **Computer/Laptop** – Runs AI and video processing.
3. **Microphone (Optional)** – For future voice commands.

### ****Software Components:****

1. **Python** – Main programming language.
2. **OpenCV** – For real-time video processing.
3. **MediaPipe** – For hand gesture and eye tracking.
4. **PyQt5** – For GUI and media player controls.
5. **QMediaPlayer** – Handles video playback.

**PROGRAM**

import sys

import cv2

import numpy as np

import mediapipe as mp

from PyQt5.QtWidgets import QApplication, QWidget, QPushButton, QSlider, QVBoxLayout, QLabel, QFileDialog, QHBoxLayout

from PyQt5.QtMultimedia import QMediaPlayer, QMediaContent

from PyQt5.QtMultimediaWidgets import QVideoWidget

from PyQt5.QtCore import Qt, QUrl, QTimer

class VideoPlayer(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Initialize Media Player

self.mediaPlayer = QMediaPlayer(None, QMediaPlayer.VideoSurface)

# Video Widget

self.videoWidget = QVideoWidget()

# Buttons

self.playButton = QPushButton("Play")

self.stopButton = QPushButton("Stop")

self.openButton = QPushButton("Open File")

self.volumeSlider = QSlider(Qt.Horizontal)

self.volumeSlider.setRange(0, 100)

self.volumeSlider.setValue(50)

self.seekBar = QSlider(Qt.Horizontal)

self.seekBar.setRange(0, 0)

# Layout Setup

controlLayout = QHBoxLayout()

controlLayout.addWidget(self.playButton)

controlLayout.addWidget(self.stopButton)

controlLayout.addWidget(self.openButton)

controlLayout.addWidget(QLabel("Volume"))

controlLayout.addWidget(self.volumeSlider)

layout = QVBoxLayout()

layout.addWidget(self.videoWidget)

layout.addWidget(self.seekBar)

layout.addLayout(controlLayout)

self.setLayout(layout)

# Event Listeners

self.playButton.clicked.connect(self.play\_video)

self.stopButton.clicked.connect(self.stop\_video)

self.openButton.clicked.connect(self.open\_file)

self.volumeSlider.valueChanged.connect(self.set\_volume)

self.seekBar.sliderMoved.connect(self.set\_position)

self.mediaPlayer.setVideoOutput(self.videoWidget)

self.mediaPlayer.positionChanged.connect(self.position\_changed)

self.mediaPlayer.durationChanged.connect(self.duration\_changed)

# Eye & Hand Detection

self.timer = QTimer()

self.timer.timeout.connect(self.process\_frame)

self.timer.start(100)

self.eye\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_eye.xml')

self.cap = cv2.VideoCapture(0)

self.eye\_detected = False

self.mp\_hands = mp.solutions.hands

self.hands=self.mp\_hands.Hands(min\_detection\_confidence=0.5,min\_tracking\_confidence=0.5)

self.mp\_draw = mp.solutions.drawing\_utils

def open\_file(self):

filename, \_ = QFileDialog.getOpenFileName(self, "Open Video")

if filename:

self.mediaPlayer.setMedia(QMediaContent(QUrl.fromLocalFile(filename)))

self.playButton.setEnabled(True)

def play\_video(self):

if self.mediaPlayer.state() == QMediaPlayer.PlayingState:

self.mediaPlayer.pause()

else:

self.mediaPlayer.play()

def stop\_video(self):

self.mediaPlayer.stop()

def set\_volume(self, value):

self.mediaPlayer.setVolume(value)

def set\_position(self, position):

self.mediaPlayer.setPosition(position)

def position\_changed(self, position):

self.seekBar.setValue(position)

def duration\_changed(self, duration):

self.seekBar.setRange(0, duration)

def process\_frame(self):

ret, frame = self.cap.read()

if not ret:

return

height, width, \_ = frame.shape

# Eye Detection for Play/Pause

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

eyes = self.eye\_cascade.detectMultiScale(gray, 1.2, 5)

if len(eyes) > 0:

if not self.eye\_detected:

self.mediaPlayer.play()

self.eye\_detected = True

else:

if self.eye\_detected:

self.mediaPlayer.pause()

self.eye\_detected = False

# Draw control regions

cv2.rectangle(frame, (50, 50), (250, 250), (0, 255, 0), 2) # Volume Control Box

cv2.putText(frame, "Volume", (90, 40), cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 255, 0), 2)

cv2.rectangle(frame, (width-250, 50), (width-50, 250), (0, 0, 255), 2) # Seek Control Box

cv2.putText(frame, "Forward", (width-200, 40), cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

frame\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

results = self.hands.process(frame\_rgb)

if results.multi\_hand\_landmarks:

for hand\_landmarks in results.multi\_hand\_landmarks:

self.mp\_draw.draw\_landmarks(frame,hand\_landmarks, self.mp\_hands.HAND\_CONNECTIONS)

# Get hand position

wrist = hand\_landmarks.landmark[self.mp\_hands.HandLandmark.WRIST]

index\_tip = hand\_landmarks.landmark[self.mp\_hands.HandLandmark.INDEX\_FINGER\_TIP]

hand\_x, hand\_y = int(index\_tip.x \* width), int(index\_tip.y \* height)

# Volume Control (Left Box)

if 50 < hand\_x < 250 and 50 < hand\_y < 250:

volume = int((1 - (hand\_y - 50) / 200) \* 100)

self.mediaPlayer.setVolume(max(0, min(100, volume)))

self.volumeSlider.setValue(volume)

# Seek Control (Right Box)

elif width-250 < hand\_x < width-50 and 50 < hand\_y < 250:

position = int((hand\_x - (width-250)) / 200 \* self.mediaPlayer.duration())

self.mediaPlayer.setPosition(position)

cv2.imshow("Gesture Control", frame)

cv2.waitKey(1)

def closeEvent(self, event):

self.cap.release()

cv2.destroyAllWindows()

event.accept()

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

player = VideoPlayer()

player.setWindowTitle("AI Gesture-Controlled Video Player")

player.resize(800, 600)

player.show()

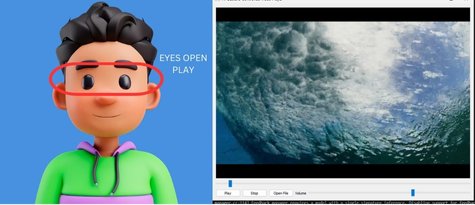
sys.exit(app.exec\_())

**RESULT**

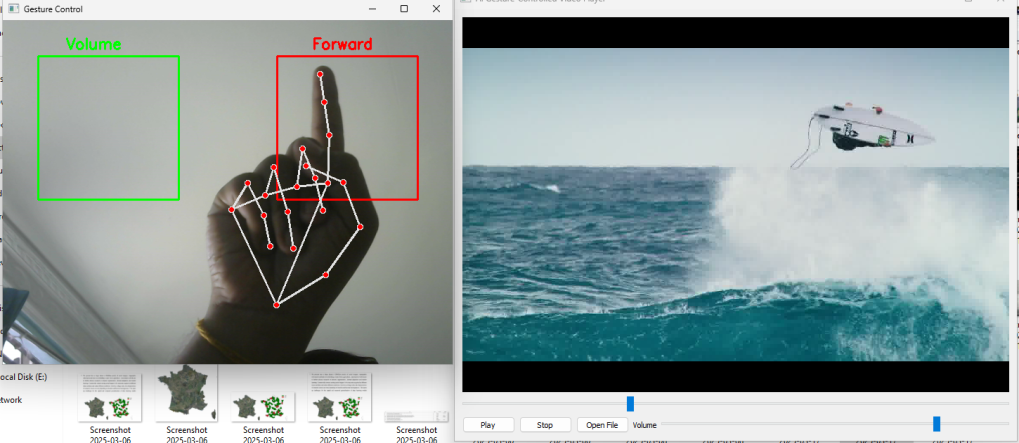
The AI Gesture-Controlled Video Player successfully enables hands-free control using eye tracking for play/pause and hand gestures for volume and seeking. It works in real time with a basic webcam, making it cost-effective and accessible. Future improvements can enhance gesture accuracy, lighting adaptability, and platform integration.

**OUTPUT SCREENSHOTS**

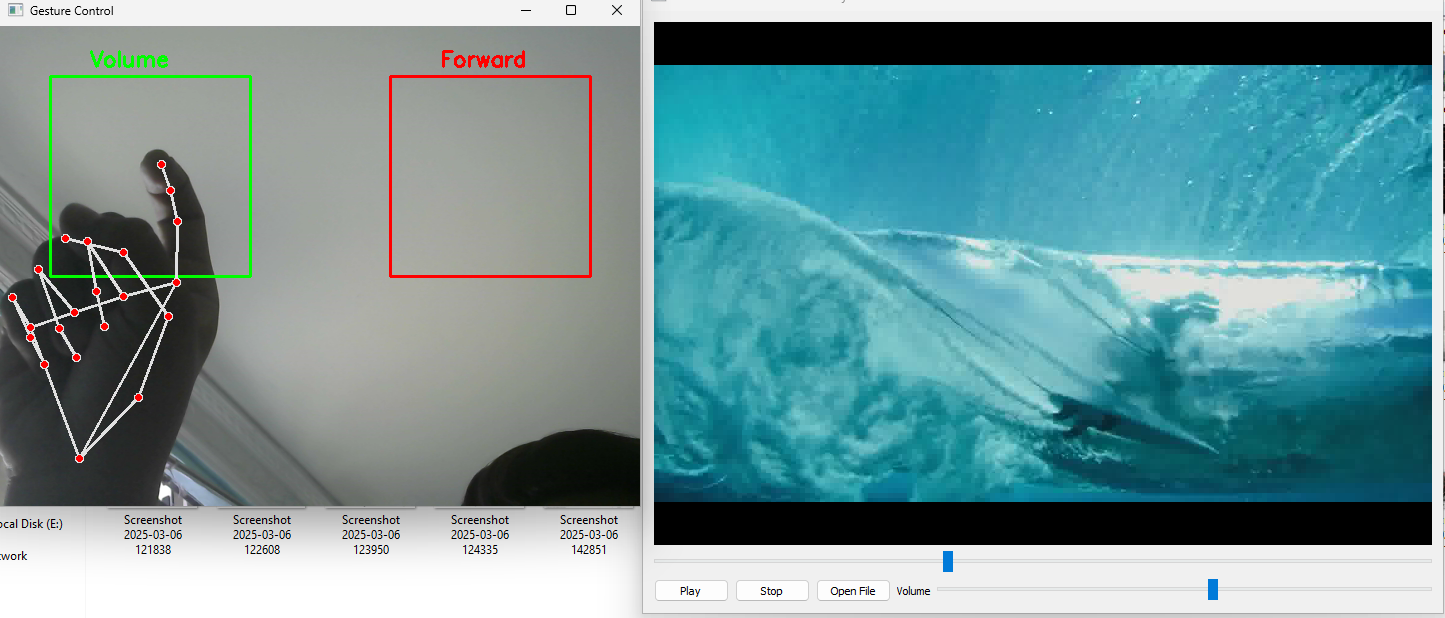
Play/Pause by Eye Detection



Forward/Backward controlled by hand gestures



Volume Up/Down controlled by hand gestures



**CHAPTER 9**

**CONCLUSION**

The Hands-Free Video Control System makes watching videos easier by using eye and hand gestures instead of touch or voice commands. This helps people with disabilities, those in rehabilitation, and anyone who needs a touch-free way to control videos.Unlike existing methods, this system is more accurate, easy to use, and does not require extra hardware. It can be useful in healthcare, smart devices, and entertainment. In the future, we can improve it by adding voice control, support for more devices, and AR/VR features. This project shows how AI can make technology more helpful and accessible for everyone.

**CHAPTER 10**

**REFERENCES**

1. **"A Survey on Hand Gesture Recognition Techniques, Methods, and Applications"** – Mitra, S., & Acharya, T. IEEE Transactions on Systems, Man, and Cybernetics
2. **"Deep Learning for Eye-Tracking-Based Human-Computer Interaction: A Review"** – Zhang, X., & Lyu, Y. ACM Computing Surveys
3. **"Real-Time Hand Gesture Recognition Using Deep Learning"** – Molchanov, P., Gupta, S., & Kim, K. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
4. **"AI-Powered Assistive Technologies for People with Disabilities"** – Sharma, A., & Patel, R. Journal of Assistive Technologies
5. **"Computer Vision-Based Gesture Control: Applications in Smart Environments"** – Liu, H., & Wang, Y. Sensors Journal
6. **"Gaze-Based Interaction in Virtual and Augmented Reality"** – Holmqvist, K., & Blignaut, P. Journal of Virtual Reality & Intelligent Hardware
7. **"The Role of AI and Computer Vision in Enhancing Accessibility"** – Chen, J., & Smith, R. AI & Society Journal